How to Help to Ask for Help? Help Request Prompt Structure Influence on Help Request Quantity and Course Retention

Sami Sarsa
Department of Computer Science
Aalto University
Espoo, Finland
sami.sarsa@aalto.fi

Jesper Pettersson
Department of Computer Science
Aalto University
Espoo, Finland
jesper.pettersson@aalto.fi

Arto Hellas
Department of Computer Science
Aalto University
Espoo, Finland
arto.hellas@aalto.fi

***Abstract*—*Full research paper*—Feedback and support are at the core of efficient learning. While the effect of feedback has been explored in a multitude of studies, the effect of asking for help and the effect of how that help is asked for is an under-explored area. In this work, we present the results of a randomized controlled trial organized in an introductory programming course for lifelong learners. In the study, we explored the usefulness of different types of help prompts used to guide learners asking for help. Gauging the effect of different combinations of three prompts used to scaffold writing out the help request, (1) "Describe the issue with your program" , (2) "Explain how your program works", and (3) "What have you tried to do to resolve the issue", we study how prompts influence learners' behavior. Using log data collected from the online platform where the prompts were explored, we study how the prompts affect whether learners end up sending a help request instead of simply considering to send one, how the scaffolding prompts affect whether learners send further help requests, and whether the questions have an effect on course retention. Our results show that the help request prompts have an impact on whether learners end up actually writing a help request, which in turn also influences whether the learner will ask for help on another occasion. Further, we observe that help request prompts may have an effect on whether learners figure out a solution to a problem on their own. Alarmingly, we also observe that the prompts can affect how far learners go in a course. Based on our results, we advise teachers and researchers to pay attention into how learners are guided into asking for help.**

***Index Terms*—feedback, help request, asking for help**

I. Introduction

Asking for help can be hard. Asking for help implies admitting that one cannot – or is not willing to – solve an issue on one's own. This in turn could be linked with feeling incompetent (or showing incompetence), something that many strive to avoid (e.g. [1]–[3]). Asking for help also means giving up control and relying on others, which could be hard due to, e.g., fear of being rejected [3], or a desire of being autonomous [4]. Help seeking – or help seeking avoidance – behaviors are intertwined with aspects emphasized in class such as mastery and performance [1], approaches to studying [5], as well as with a range of motivational factors and attitudes [6].

Given the many benefits of asking for help and receiving help and feedback, including help leading to more completed coursework [7] and better course outcomes [8], it is crucial to instill the notion that it is perfectly acceptable – and preferable – to ask for help when needed. Indeed, help seeking is a strategy that is a part of self-regulated learning [2], [9]. The strategy is learned over time [10] and evolves with the ability to monitor own understanding [11] – one could also view help seeking as an alternative (or as a complement) to individual problem solving [12], where one taps into external resources [13]. Before asking for help, however, one needs to be aware of needing help (and of the possibility of being able to ask for help) [14].

As asking for help can be hard, how then could we help in asking for help? In this work, we set out to explore the utility of help request prompts used to scaffold learners as they are writing a help request. The work was conducted on an online E-Book and learning platform used for teaching principles of computer science through a variety of (automatically assessed) programming exercises. Whenever there were problems with exercises submitted for automated assessment, the platform showed a button prompting the learners to ask for help. Pressing the button opened a dialog with prompts such as "Describe the issue with your program" – we call these *scaffolding prompts* – that were intended to help the learners to write their help requests. The prompts shown to learners were randomly picked from a pool of prompt options, allowing us to study the effect of the prompts and their combinations on learner behavior.

Our research questions for the present work are:

RQ1 How do the scaffolding prompts influence the amount of submitted help requests?

RQ2 How do the scaffolding prompts influence the resolving of the submitted help requests?

RQ3 How do the scaffolding prompts influence whether a learner who opens the help request dialog submits a help request?

RQ4 How do the scaffolding prompts influence course retention?

II. Background

A. Automated assessment and feedback in programming

A wide variety of tools are used for providing feedback in learning programming. These tools range from automated assessment systems (for reviews, see e.g. [15]–[18]) and systems used to illustrate how programs work [19], [20]. While automated assessment systems originally focused on functional correctness of programs (cf. [15]–[18]), systems have since included functionality for assessing code quality [21], assessing student-written tests [22], and functionality for providing step-by-step guidance [23].

Plenty of evidence for the use of automated assessment systems [24] already exists, yet excessive reliance on them can be detrimental [25]. The availability of automated assessment systems can lead to students attempting to game the system instead of seeking to understand the problems in their programs [25], [26], even to the extent that students may be discouraged from testing programs on their own [27]. Some of the problems arising from the use of automated assessment may be due to students still learning how to learn, and not being aware of what they know and what they do not know [28]. Indeed, practising problem-solving strategies and aiding in understanding of a given problem can improve students' performance [28]–[30].

While the previous examples have focused primarily on automated assessment systems, there are also systems that help providing written feedback to students. At one end, these systems provide teachers easy access to rubrics that help in the grading process and provide ready-made template texts that can be adjusted on the fly [31], [32]. On the other end, these systems make it easier to group similar submissions together to allow creating and providing feedback to groups of students [33]–[35]. In practice, one can argue that there is still a need for a human in the loop – or at least in some parts of it – as aspects still remain in assessment (and in providing feedback) that haven't been automated, or if they have, the automation needs monitoring and human interventions [36].

B. Help seeking in programming

Sending an exercise into an automated assessment system can serve a dual purpose. On one hand, students' solution to exercises are graded, but on the other hand, students also receive feedback on the correctness of their solution. In practice, this means that automated assessment systems are used both as systems for grading and for seeking help. Since automated assessment systems can be misused [25], [26], as discussed previously, this dual purpose has its downsides; it may be, for example, that students might be seeking for help but by chance have reached a correct solution, and get the feedback that their program is correct. We see a need to distinguish help-seeking and feedback-seeking functionality from grading, which could improve students' understanding of what they are attempting to do and increase their awareness of their present knowledge, something that – or more precisely their lack of it – has been also priorly raised as a concern [28], [37].

The natural collaboration and discussion opportunities differ in programming courses depending on how the courses are organized. To provide a few examples, the pedagogical approaches can emphasize collaboration and making things visible [38] that serve as a natural way to facilitate discussion, and the pedagogical approach can also emphasize working in labs while being guided by others [39]. One could, also, use office hours for asking for help [40].

All students typically cannot attend instruction, so the basic (technical) help-seeking functionality offered in courses often boils down to the use of dedicated discussion systems that make it easier to ask questions about the course work. As asking for help is notoriously difficult [1]–[4], [41], one might opt to attempt to resolve problems on their own, and perhaps even resort to plagiarism [42]. This behavior is to some extent linked with students' self-efficacy perceptions, which influences how students behave when facing challenges [43].

We acknowledge that there are multiple sources wherefrom students can seek help, many of which are external to the opportunities offered by the course [37], [44]. When seeking help in a programming course, students can ask for help from their peers and search for information from a variety of online services [37], [44]. Help-seeking functionality has also been built into some learning systems, for example, in the form of hint functionality [45]. Regardless, there is a need to further develop tools to better support self-regulated learning and help-seeking [46].

C. Supporting seeking of help

Given the difficulty of asking for help and the benefits of receiving help, the starting point should be that students are encouraged to seek help – this already increases the likelihood of seeking help [7]. When considering systems used for teaching programming, additional emphasis should be put into building functionality that allows fast help-seeking [47]. This could manifest, e.g., through built-in in hint functionality [45] that somehow moderates unproductive help-seeking such as asking for help when help is not needed [48]. Or, the functionality could manifest through other ways that allow asking help, such as easily accessible help request prompts. Although help-seeking is a self-regulated learning strategy [2], [9], one could view it also as a skill that can be honed.

When considering what to practice in help-seeking, one should consider what effective help-seeking is like. In practice, effective help-seeking is embodied through the ability to ask precise questions, to persist in seeking help, and to apply the received explanations [49]. That is, forming a question is a part of learning to seek help. One way to improve help-seeking would be to suggest ways to ask questions [50], or to even provide questions that act as scaffolding [51]. Such questions could target different parts of the problem; one could, for example, prompt for the expected behavior of a program and the present behavior of a program [51].

D. Research gap

In the present work, we are studying the effects of (1) separating help-seeking functionality from automated assessment, and (2) providing different prompts – *scaffolding prompts* – that are used to help writing help requests. In particular, we are looking into to what extent different prompts help and to what extent different prompts do not help – one instance of prompts not helping could manifest through having too complex prompts or too many prompts. At the minimum, this can influence response times [52] and potentially also the eagerness to answer, despite prior research from other contexts suggesting that survey length has little effect on response rate [53].

III. Methodology

A. Context

This study was conducted on an online platform used for offering open online lifelong learning courses organized by Aalto University. The platform offers four courses: (1) introduction to programming; (2) data and information; (3) internet and browser applications; and (4) mobile application development. Each course is worth 2 ECTS[1], corresponding to approximately 50 to 60 study hours per course.

Each course has programming exercises that are intertwined with the online learning materials and that are used to introduce and to provide a broader context to course concepts. The programming exercises are worked on in an integrated programming environment. The exercises are automatically assessed, and the learners[2] can proceed in the courses at their own pace. The grading of the courses is based on completed exercises; to pass a course, one must complete at least 90% of the available exercises.

B. Platform setup

Whenever a programming exercise is shown in the materials, a button with the text "Request help" is displayed next to a button used to submit the programming exercise for automated assessment. For each programming exercise, the "Request help" button is at first inactive. The button becomes active when a learner submits the programming exercise and the automated assessment system highlights issues with the solution. After clicking the activated "Request help" button, a modal dialog is opened. The button is again deactivated when the learner has solved the exercise. The submitted help requests are answered by the course staff. The process of requesting help in the platform is illustrated in Figure 1.

When a learner clicks the "Request help" button for the first time, the learner is assigned into an experiment group. Each group has a *help request template* that defines which scaffolding prompts are shown. After a learner has been assigned into an experiment group, the group does not change.

[1]European Credit Transfer System

[2]We refer to the course participants as *learners* instead of the more commonly used term *students* as they are not degree students.

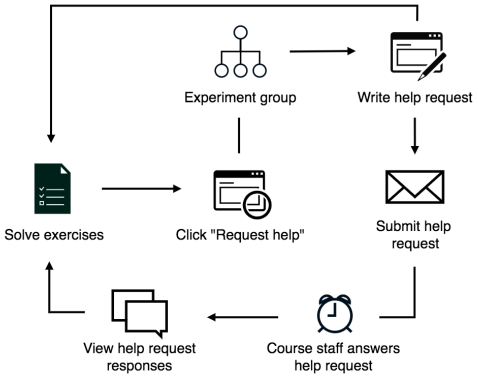

Fig. 1. Life-cycle of the learner within the context of solving exercises and seeking feedback using help requests.

The modal dialog (that appears when the "Request help" button is clicked) displays the code files for the latest (failed) programming exercise submission in a read-only format, as well as the scaffolding prompts defined in the help request template associated with the experiment group. The text-fields linked to each scaffolding prompt are defined as required, i.e. the learner must enter an answer for each scaffolding prompt to be able to submit the help request. The platform notifies the learner when help requests have been answered; the learner can view answered and unanswered help requests on the platform on a separate page.

Once a help request has been created it is assigned a "not resolved" state. Changing the state of a help request is accomplished in two ways: (1) The learner manually marking a help request as resolved; or (2) the learner solving the exercise for which the help request was created. Course staff cannot mark help requests as resolved – when answering help requests, the course staff only sees help requests that have not yet been answered and that have not yet been resolved. The help requests are displayed anonymously on the page visible to course staff.

C. Help request templates

In the present study, we used three scaffolding prompts which are as follows.

1) Describe the issue with your program
2) Explain how your program works
3) What have you tried to do to resolve the issue?

From these prompts, we created four help request templates that each have a unique combination of the above three scaffolding questions. These combinations are outlined in Table I.

D. Experiment and data

We conducted the experiment as as a randomized experiment, where learners – when clicking the "Request help"-button on the platform for the first time – were randomly assigned into one of four groups, where each group was assigned one of the help request templates outlined in Table I. Once a learner was assigned into one of the groups, the group

TABLE I
HELP REQUEST TEMPLATES AND THEIR RESPECTIVE SCAFFOLDING PROMPT COMBINATIONS

Template / group	Prompt 1 Describe..	Prompt 2 Explain..	Prompt 3 What have you..
1	Yes	Yes	Yes
2	Yes	No	No
3	Yes	Yes	No
4	Yes	No	Yes

could not be changed, and the learner always received the same questions as defined in the help request template whenever asking for help.

For the present study, we collected log data of learners interacting with the help request functionality (i.e. opening, closing, and submitting the help request form), information on the submitted help requests (whether the help request was resolved), and learner data including learners' obtained points for course exercises as well as the group that each learner was assigned to (if they had opened the help request form). All of the data used is collected within the course platform following the ethical principles of research outlined in *Anonymous university* guidelines.

E. Analyses

To answer RQ1, RQ2 and RQ3, we display the data in tables and present and analyse the statistics related to each group of learners. More specifically, we analyse (1) how many learners open the help request dialog and how many submit help requests within each group, (2) how the numbers of total opened help request dialogs and submitted help requests differ across the groups, and (3) how resolving help requests differ across the groups.

For RQ4, we analyse the completion rates within courses and how they differ in the help request scaffolding groups. When inspecting course completion, we only include learners who have not submitted a solution to an exercise in the last 30 days. This is done to prevent new learners from being counted as learners who have not completed a course. We note that a learner may have multiple completions if they have completed exercises in multiple courses. For instance, a learner who has completed exercises in two distinct courses will have two completion rates. When we analyse the completion rates, we do not analyse the rates per learner but the rates individually.

In addition, we use pairwise Mann-Whitney U tests to evaluate the significance of distribution differences within the groups regarding the number of requests per learner, submitting an opened request, and course completion rate.

IV. RESULTS

During the study, 1938 learners had worked on at least one of the courses and roughly half of them, 957, clicked the "Request help" button. Of the 957 learners who opened the help request form at least once, around one third submitted at least one help request. On average, each learner who submitted a request, submitted 2.83 requests. A total of 84.9% of the help requests were resolved, meaning that either the learner resolved it manually or had completed the exercise related to the request. On the other hand, 15.1% of the help requests were not resolved, meaning that in 15.1% of the cases the learner did not finish the exercise associated with the help request. These statistics are shown in detail and also separately for each group in Table II.

As can be seen from Table II, both the number of requests and the number of learners who requested help vary between the groups. Most notably, group 1, the one that includes all three scaffolding prompts, has distinctively fewer requests and requesters of help than the other groups. Likewise, group 2, the group with a template with only 1 prompt boasts the largest number of requests and help requesting learners, although the difference is not large when compared to group 4. Group 3, the group with the two prompts (1) "Describe the issue with your program" and (2) "Explain how your program works" falls in between group 1 and the other groups when considering the number of requests and learners seeking help.

For a closer look on the effect of scaffolding prompts towards the number of requests submitted by learners, we performed pairwise Mann-Whitney U tests on the number of requests by the learners. The results are presented in Table III, where we can see that the differences between group 1 and the other groups 2 and 4 appear significant with p-values ranging from ~ 0.01 to ~ 0.002 although all the effect sizes should be considered small ($|r| < 0.3$). The differences the other group comparisons appear to be more likely random according to the Mann-Whitney U tests with p-values ranging from ~ 0.2 to ~ 0.6

A. Resolving help requests

Turning our attention to the percentage of resolved help requests in Table II, we see that group 2 has the highest percentage of 88%, which is closely followed by 86% in group 3. Groups 1 and 4 that both include the prompt (3) "What have you tried to do to resolve the issue?" perform the worst out of the groups with somewhat lower percentages of 83% and 82% respectively.

A similar picture can be seen in Table IV, which shows learner statistics per exercise for the different groups to account for cases where learners submit multiple help requests for a single exercise. The rate of completing an exercise for which help has been requested is akin to the rate of resolving a request. This is unsurprising as the number of help requests per exercise and learner is close to one (omitting cases where help has not been requested). When looking at resolving requests prior to receiving response (i.e. without help from course staff), group 2 has the highest ratio of help requests being resolved without help, while the ratio is the poorest for the group 1. Groups 3 and 4 fall in between, with group 4 performing marginally better in this regard.

TABLE II
HELP REQUEST STATISTICS FOR THE DIFFERENT TEMPLATES

Template	All learners	Learners with requests	Requests	Requests / Learners	Requests / Learners with requests	Resolved %
1	242	72	145	0.60	2.01	83.4%
2	232	104	299	1.29	2.88	88.0%
3	245	87	248	1.01	2.85	86.3%
4	238	95	290	1.22	3.05	82.1%
-	981	-	-	-	-	-
Total	1938	358	982	1.03	2.83	84.9%

TABLE III
PAIRWISE MANN-WHITNEY U TESTS AND EFFECT SIZES (r) REGARDING NUMBER OF REQUESTS PER LEARNER DIFFERENCES BETWEEN GROUPS.

x (U_1) →	1			2			3			4		
y (U_2) ↓	U_1	p	r	U_1	p	r	U_1	p	r	U_1	p	r
1	29282.0	1.0000	0.0000	32074.0	0.0017	0.1426	31246.0	0.2101	0.0540	32092.0	0.0103	0.1144
2	24070.0	0.0017	-0.1426	26912.0	1.0000	0.0000	26103.0	0.0756	-0.0815	26954.0	0.6157	-0.0237
3	28044.0	0.2101	-0.0540	30737.0	0.0756	0.0815	30012.5	1.0000	0.0000	30848.0	0.1982	0.0581
4	25504.0	0.0103	-0.1144	28262.0	0.6157	0.0237	27462.0	0.1982	-0.0581	28322.0	1.0000	0.0000

The group sample sizes can be seen in the "All learners" column in Table II.

TABLE IV
PER LEARNER AND EXERCISE STATISTICS FOR THE DIFFERENT HELP REQUEST TEMPLATES

Template	Mean requests per exercise	Exercise completed %	Resolved without response %
1	1.16	82.2%	22.9%
2	1.25	88.0%	29.7%
3	1.29	85.3%	23.1%
4	1.22	81.1%	25.2%

B. Opening and submitting help requests

Table II showed that plenty of learners who were assigned to a group, meaning that they opened a help request form at least once, never submitted a help request and that the amount of submitters varied between the groups. Table V further shows that the amount of opened help request forms and also the percentage of opened forms being submitted greatly varies between the groups.

TABLE V
STATISTICS OF OPENING AND SUBMITTING HELP REQUESTS FORMS FOR THE DIFFERENT TEMPLATES

Template	Opened	Submitted	Submitted / Opened
1	781	149	19.1%
2	971	303	31.2%
3	887	245	27.6%
4	986	304	30.8%

Group 1 has the least opened forms and the lowest rate of submitting an opened form, groups 2 and 4 have the most opened forms and highest submit/open rates and group 3 falls in between in both categories. This is so far similar to when comparing total requests submitted. The only major difference we find in opening a form as opposed to submitting a request form is the wider difference in submitting the help request form than opening the form across the groups. Furthermore, we can see a near identical picture in the Mann-Whitney U tests on learners' submit / open ratios seen in Table VI as with the tests on request counts in Table III.

C. Help requests and course completion

Figure 2 shows the completion rates in box plots for both all learners in the groups and for learners in the groups with at least one help request submission. For learners with at least one help request submission, we can see a clear difference in the means and medians in favor of groups 2 and 3 when compared to groups 1 and 4. In contrast, when looking at the box plots for all learners in groups, i.e. anyone who has at least opened a help request form, the means look rather similar and group 1 has the highest median.

We computed Mann-Whitney U test for significance in course completion differences in the template groups with results shown in Table VII. Therein we can see that significant ($p < 0.05$) difference between groups 1 and 2, and 2 and 4. Somewhat significant differences ($0.05 < p < 0.06$) can also be seen between groups 1 and 3, and between 3 and 4. As with the previous tests, all the effect sizes should be considered small ($|r| < 0.3$).

V. DISCUSSION

A. Effect of scaffolding prompts

The clearest observation from our results is that having fewer structured (only one) prompts correlates with the most interaction with the help request functionality, learners who eventually submit at least one help request and also good completion rates. When considering prior studies on web

TABLE VI
PAIRWISE MANN-WHITNEY U TESTS AND EFFECT SIZES (r) REGARDING SUBMIT / OPEN RATIO DIFFERENCES BETWEEN GROUPS.

x (U_1) → / y (U_2) ↓	1			2			3			4		
	U_1	p	r	U_1	p	r	U_1	p	r	U_1	p	r
1	29282.0	1.0000	0.0000	31998.0	0.0022	0.1399	31054.0	0.2723	0.0475	31995.5	0.0133	0.1110
2	24146.0	0.0022	-0.1399	26912.0	1.0000	0.0000	25863.0	0.0510	-0.0900	26878.5	0.5779	-0.0264
3	28236.0	0.2723	-0.0475	30977.0	0.0510	0.0900	30012.5	1.0000	0.0000	30971.0	0.1696	0.0623
4	25600.5	0.0133	-0.1110	28337.5	0.5779	0.0264	27339.0	0.1696	-0.0623	28322.0	1.0000	0.0000

The group sample sizes can be seen in the "All learners" column in Table II.

TABLE VII
PAIRWISE MANN-WHITNEY U TESTS AND EFFECT SIZES (r) REGARDING COURSE COMPLETION DIFFERENCES BETWEEN GROUPS.

x (U_1) → / y (U_2) ↓	1			2			3			4		
	U_1	p	r	U_1	p	r	U_1	p	r	U_1	p	r
1	6160.5	1.0000	0.0000	9189.0	0.0387	0.1498	7884.0	0.0530	0.1456	9837.0	0.1829	0.0941
2	6795.0	0.0387	-0.1498	10368.0	1.0000	0.0000	8798.5	0.8365	-0.0145	11097.0	0.4559	-0.0486
3	5880.0	0.0530	-0.1456	9057.5	0.8365	0.0145	7688.0	1.0000	0.0000	9652.0	0.5674	-0.0390
4	8145.0	0.1829	-0.0941	12231.0	0.4559	0.0486	10436.0	0.5674	0.0390	13122.0	1.0000	0.0000

The group sample sizes can be seen in the "Learners with requests" column in Table II.

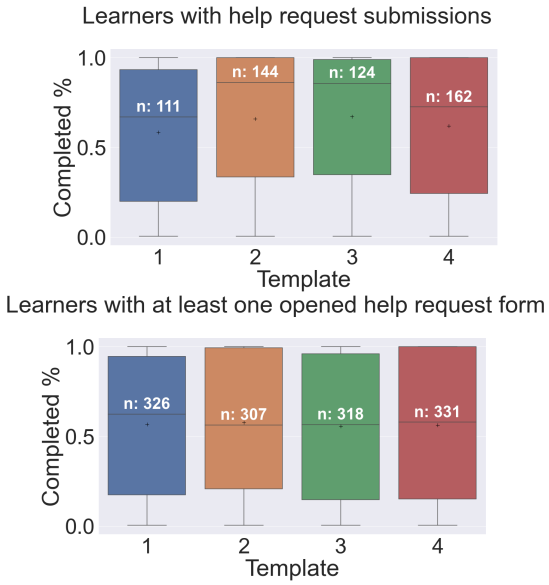

Fig. 2. Course completion rates. Sample sizes are represented in white and exceed the number of learners due to some attempting multiple courses. The "+"-sign represents the average completion rate.

surveys that have highlighted that the number of questions does not affect answer ratio [53], it is clear that requesting help differs from answering surveys. In general, the differences could be partly explained by the increased cognitive load or even by a possible detrimental effect to the self-efficacy of learners when they feel they are required to voice out issues related to their code that they have trouble understanding.

We find an interesting distinction on the effect of code explanation (prompt 2) when compared to the approaches tried so far (prompt 3). Including the former appears to affect negatively the amount help requests, while the latter seems to have the same effect on course completion rate. The prompt 2 (code explanation) could reveal a lack of skills or make the learners feel incompetent, which some learners tend to avoid [1]–[3]. Similarly, the prompt 3 (approaches tried so far) could reveal a lack of meta-cognitive skills [28], [29] which could manifest in less systematic studying and hence larger dropouts.

B. Looking beyond the data

For now we have focused on the existence and extent of differences between the groups, but we should also understand what are the implications of these differences. While differences in completion rate are straightforward to interpret as a higher completion rate is naturally better and should be preferred, the differences in request counts and submit / open ratios are less straightforward. A small request count could indicate that learners are turned off from requesting help (a bad thing) or it could also indicate that the prompts have helped the learners to better solve problems on their own (a good thing). Similarly, a low submit open ratio could indicate that learners feel that answering the prompts is too demanding (a bad thing), or that the prompts help them solve their issue dismissing the need to submit the help request (a good thing).

In an ideal situation, the prompt would lead to the learner thinking about the problem and consequently realizing the underlying issue during the process of forming explanations of the code. This would be similar to the use of rubber duck debugging [54] where issues are described to an inanimate object, which can help in resolving the issue. The prompts provided in the platform can act in a similar way, although the need to explain the issue is invoked by the platform and not the learner. In general, at the moment, we do not know whether closing the dialog and not submitting a help request is a signal of understanding the issue, or a signal of not wishing to write in the details. Based on the relatively low ratio of

learners who submit a help request out of those who open a help request form, it is evident that something happens. One avenue for future research would be to explicitly ask why the prompt was closed. If it is because of not wishing to write the details, the course would benefit from code explanation questions – something that the course currently is missing – as practicing with code explanation questions could help with explaining the code while asking for help, in addition to the benefits related to program comprehension [28]–[30].

C. Some differences between courses and learners

While the present analysis focused on aggregate statistics over the courses offered by the online platform, we acknowledge that there were some small differences between the courses in terms of the use of the help request functionality. In particular, in the last course that focused on mobile application development, the fourth template that asked the learners to describe the problem and to explain what they had tried so far led to a marginally higher completion rate than the third template. Due to a small number of participants in the mobile application development course, we chose not to look into this further in the present study. This observation however does provide some additional evidence on the suggestion that the help request prompts – and more broadly instructional strategies – should be adjusted to match the learners present knowledge and the course [55]. In the case of a larger program (which was often the case in the mobile application development course), it is possible that asking to explain what the program does could simply feel as a too large of a task. Yet, it is also possible that the learners at that point already kind-of know what the program is doing and feel that explicitly explaining the behavior in written form is irrelevant and therefore frustrating. The latter possibility would be in line with the expertise reversal effect [55], which states that some materials and learning activities useful for novices can be detrimental for more experienced learners.

When looking into the written help requests, we did observe that some learners skipped some of the prompts, answering them very briefly or even writing only a dot (leaving the text-field empty was not allowed). Similarly, we did observe that some learners had challenges in distinguishing how their program worked and how the program should work (i.e. describing the expected outcome instead of their code). While a qualitative analysis of the help requests was out of scope for our present study as we looked into the higher level effects between the groups, this is something that would be interesting to study further in the future. In practice, for example, if some of the learners skipped some scaffolding prompts, it might even be that some prompt combinations are worse than what our present analysis reveals.

While the online platform used in the experiment is used for providing online lifelong learning courses, we were able to discuss with some participants who used the platform informally at events organized by the *Anonymous University*. When discussing about the help seeking functionality and the courses in general, we encountered a learner who stated that they never submitted work to be graded so as to not burden the teachers. Due to the significant use of automated assessment systems in teaching computer science, this statement was rather surprising to us, and it did leave us to wonder whether some of the learners also thought similarly, i.e. that they would not submit a help request in order to not burden the teachers. It is a good question whether the "Request help" button that becomes active when submitting a failing exercise should be somehow emphasized more, or whether clicking the button and submitting a help request should even be rewarded in some way, to guide learners towards asking for more help.

D. Limitations of work

This work comes with a number of limitations, which we discuss next. Firstly, although learners can only access the help request dialog if the submission to the exercise that they currently work on fails, we do not know whether the learners had a problem when they opened the help request dialog. It is possible that some learners simply wanted to see what would happen if they would click the button. Similarly, we do not know whether seeing the help request prompts helped some learners resolve the issue on their own.

Secondly, we have not conducted a deeper analysis of the different programming exercises (and courses) where learners have sought help. Considering the differences in the programming exercises (and courses) in the analysis could provide more insight into the level of difficulty that learners were facing; supporting insight could also be gained from the analysis of the written help requests and the associated code submissions, from learner demographic data, and also from the responses from the course staff and the time that it took to write them. For the present analyses that looked for trends between the groups, such data was scoped out.

Thirdly, our experiment did not utilize a control group that would not see the help request button at all. Due to ethical considerations we chose to provide a template to all learners in order avoid a situation where some learners would have no functionality for asking for help. We acknowledge that there exists prior research that has shown that not all features are beneficial over not having those features at all (e.g. [56]) but due to the voluntary aspect of the help request functionality – i.e. learners could decide whether they asked for help – we believe that the net effect was positive.

VI. Conclusion

In this article, we presented the results of a research study where we associated a "Request help" button with each programming exercise. The button became active whenever a learner submitted a programming exercise for which the automated tests did not pass. Clicking on the button associated the learner with one of four groups, where each group had a varying combination of scaffolding prompts that provided structure to asking for help and that were intended to help writing the help requests. The scaffolding prompts explored were (1) *Describe the issue with your program*, (2) *Explain how your program works*, and (3) *What have you tried to*

do to resolve the issue?. The first group had the scaffolding prompts 1-3, the second group had the scaffolding prompt 1, the third group had the scaffolding prompts 1-2, and the fourth group had the scaffolding prompts 1 and 3. As a summary, our research questions and their answers are as follows.

RQ1 How do the scaffolding prompts influence the amount of submitted help requests? **Answer:** *We observed that the scaffolding prompts influence the number of submitted help requests. Out of the explored scaffolding prompt combinations, the group with the prompt that solely asked the learner to describe the issue with their program most often led to submitted help requests. The second best group in this regard was the fourth group, where learners were asked to describe the issue with their program and to describe what they had already tried to resolve the issue. In general, it seems that asking the learner to explain how their program works that was present in the first and third groups led to fewer submitted help requests, while the fewest help requests were submitted by the learners in the group that had all of the three scaffolding prompts.*

RQ2 How do the scaffolding prompts influence the resolving of the submitted help requests? **Answer:** *Again, the second group where learners were asked to describe the issue with their program ranked the highest in terms of (a) the help requests being more often resolved without a response from the course staff (i.e. the learner solved the exercise on their own before a response or marked the help request as resolved) and (b) the help requests being resolved overall. The first group with all the scaffolding prompts most likely – out of all of the groups – did not resolve the help request without response.*

RQ3 How do the scaffolding prompts influence whether a learner who opens the help request dialog submits a help request? **Answer:** *Again, the second group where learners were asked to describe the issue with their program was the most likely one to lead to learners creating at least one help request – 44.8% of the learners in the group submitted at least one help request. The group with all three scaffolding prompts was the least likely to lead to learners creating at least one help request – 29.8% of the learners in the group submitted at least one help request. The proportions of learners submitting a help request in the third and fourth group were 35.5% and 39.9% respectively, falling in between the first and second group. Moreover, when considering the total number of times that the help request prompts were opened and a help request was submitted, there are also considerable differences between the groups. Similarly to opening the help request dialog and submitting at least one help request, the second group had the highest submit / open ratio, where out of the instances where the help request dialog was opened, a help request was submitted in 31.2% of the instances. The lowest submit / open ratio was observed with the group one, 19.1%. The submit / open ratios for the groups three and four fell again between the groups one and two, being 27.6% and 30.8% respectively.*

RQ4 How do the scaffolding prompts influence course retention? **Answer:** *When considering the groups in general (i.e. based on students who opened the help request dialog but did not necessarily submit a help request), the average course completion rates do not considerably differ between the groups. However, when focusing on the groups that submitted at least one help request, there are significant differences. The groups two and three fare somewhat better than the group one in terms of course completion, while the group one and four were similar (i.e. somewhat poorer).*

Overall, our work shows that the prompts that guide learners to ask for help influence whether and how often the learners actually write and submit a help request. The prompts also influence whether the learners resolve the issues on their own as well as on whether the issues are resolved in general. Further, there is an indication of the prompts influencing course retention – at least in the populations that end up submitting at least one help request. This work highlights the need to carefully consider how learners are guided to ask for help in online environments.

While the present work focused on higher level statistics between the groups, ignoring the potential effects from differences in exercises, courses, and student demographics, we are looking into those aspects as a part of our future work. Further, we are considering a qualitative analysis of the help requests, and we are looking into encouraging learners into writing help requests, potentially adding gamification elements into this, as as well as looking into involving the learners in answering help requests made by other learners.

REFERENCES

[1] S. A. Karabenick, "Perceived achievement goal structure and college student help seeking." *J of educational psychology*, vol. 96, no. 3, 2004.

[2] A. M. Ryan, P. R. Pintrich, and C. Midgley, "Avoiding seeking help in the classroom: Who and why?" *Educational Psychology Review*, vol. 13, no. 2, pp. 93–114, 2001.

[3] D. Seamark and L. Gabriel, "Barriers to support: a qualitative exploration into the help-seeking and avoidance factors of young adults," *British J. of Guidance & Counselling*, vol. 46, no. 1, 2018.

[4] R. Butler, "Determinants of help seeking: Relations between perceived reasons for classroom help-avoidance and help-seeking behaviors in an experimental context." *J. of Educ. Psychology*, vol. 90, no. 4, 1998.

[5] M. Nelimarkka and A. Hellas, "Social help-seeking strategies in a programming MOOC," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 2018, pp. 116–121.

[6] A. M. Ryan and P. R. Pintrich, ""should i ask for help?" the role of motivation and attitudes in adolescents' help seeking in math class." *Journal of educational psychology*, vol. 89, no. 2, p. 329, 1997.

[7] C. Pellegrino, "Does telling them to ask for help work?: Investigating library help-seeking behaviors in college undergraduates," *Reference and User Services Quarterly*, vol. 51, no. 3, pp. 272–277, 2012.

[8] M. Micari and S. Calkins, "Is it ok to ask? the impact of instructor openness to questions on student help-seeking and academic outcomes," *Active Learning in Higher Education*, vol. 22, no. 2, pp. 143–157, 2021.

[9] B. J. Zimmerman and M. Martinez-Pons, "Construct validation of a strategy model of student self-regulated learning." *Journal of educational psychology*, vol. 80, no. 3, p. 284, 1988.

[10] S. G. Paris and R. S. Newman, "Development aspects of self-regulated learning," *Educational psychologist*, vol. 25, no. 1, pp. 87–102, 1990.

[11] S. N.-L. Gall, "Help-seeking behavior in learning," *Review of research in education*, vol. 12, no. 1, pp. 55–90, 1985.

[12] S. Nelson-Le Gall, "Help-seeking: An understudied problem-solving skill in children," *Developmental review*, vol. 1, no. 3, 1981.

[13] M. Cole and I. Traupmann, "Comparative cognitive research: Learning from a learning disabled child," in *Aspects of the development ofcompetence (Minnesota Symposia on childpsychology, Vol. 14)*, 1981.

[14] H. Van Der Meij, "Student questioning: A componential analysis," *Learning and individual Differences*, vol. 6, no. 2, pp. 137–161, 1994.

[15] K. M. Ala-Mutka, "A survey of automated assessment approaches for programming assignments," *Computer science education*, vol. 15, no. 2, pp. 83–102, 2005.

[16] C. Douce, D. Livingstone, and J. Orwell, "Automatic test-based assessment of programming: A review," *Journal on Educational Resources in Computing (JERIC)*, vol. 5, no. 3, pp. 4–es, 2005.

[17] P. Ihantola, T. Ahoniemi, V. Karavirta, and O. Seppälä, "Review of recent systems for automatic assessment of programming assignments," in *Proceedings of the 10th Koli calling international conference on computing education research*, 2010, pp. 86–93.

[18] J. C. Paiva, J. P. Leal, and Á. Figueira, "Automated assessment in computer science education: A state-of-the-art review," *ACM Transactions on Computing Education (TOCE)*, 2022.

[19] S. Fincher, J. Jeuring, C. S. Miller, P. Donaldson, B. Du Boulay, M. Hauswirth, A. Hellas, F. Hermans, C. Lewis, A. Mühling *et al.*, "Notional machines in computing education: The education of attention," in *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*, 2020, pp. 21–50.

[20] P. E. Dickson, N. C. Brown, and B. A. Becker, "Engage against the machine: Rise of the notional machines as effective pedagogical devices," in *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, 2020, pp. 159–165.

[21] H.-M. Chen, W.-H. Chen, and C.-C. Lee, "An automated assessment system for analysis of coding convention violations in java programming assignments." *J. Inf. Sci. Eng.*, vol. 34, no. 5, pp. 1203–1221, 2018.

[22] S. H. Edwards, "Teaching software testing: automatic grading meets test-first coding," in *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, 2003, pp. 318–319.

[23] A. Vihavainen, T. Vikberg, M. Luukkainen, and M. Pärtel, "Scaffolding students' learning using test my code," in *Proc. of the 18th ACM conference on Innovation and technology in computer science education*, 2013.

[24] R. S. Pettit, J. D. Homer, K. M. McMurry, N. Simone, and S. A. Mengel, "Are automated assessment tools helpful in programming courses?" in *2015 ASEE Annual Conference & Exposition*, 2015, pp. 26–230.

[25] E. Baniassad, L. Zamprogno, B. Hall, and R. Holmes, "Stop the (autograder) insanity: regression penalties to deter autograder overreliance," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 2021, pp. 1062–1068.

[26] R. Baker, J. Walonoski, N. Heffernan, I. Roll, A. Corbett, and K. Koedinger, "Why students engage in "gaming the system" behavior in interactive learning environments," *Journal of Interactive Learning Research*, vol. 19, no. 2, pp. 185–224, 2008.

[27] K. Buffardi and S. H. Edwards, "Reconsidering automated feedback: A test-driven approach," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 2015, pp. 416–420.

[28] J. Prather, R. Pettit, K. McMurry, A. Peters, J. Homer, and M. Cohen, "Metacognitive difficulties faced by novice programmers in automated assessment tools," in *Proceedings of the 2018 ACM Conference on International Computing Education Research*, 2018, pp. 41–50.

[29] J. Prather, R. Pettit, B. A. Becker, P. Denny, D. Loksa, A. Peters, Z. Albrecht, and K. Masci, "First things first: Providing metacognitive scaffolding for interpreting problem prompts," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 2019.

[30] A. Vihavainen, C. S. Miller, and A. Settle, "Benefits of self-explanation in introductory programming," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 2015, pp. 284–289.

[31] T. Ahoniemi, E. Lahtinen, and T. Reinikainen, "Improving pedagogical feedback and objective grading," in *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, 2008, pp. 72–76.

[32] T. Auvinen, V. Karavirta, and T. Ahoniemi, "Rubyric: an online assessment tool for effortless authoring of personalized feedback," *ACM SIGCSE Bulletin*, vol. 41, no. 3, pp. 377–377, 2009.

[33] E. L. Glassman, J. Scott, R. Singh, P. J. Guo, and R. C. Miller, "OverCode: Visualizing variation in student solutions to programming problems at scale," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 22, no. 2, pp. 1–35, 2015.

[34] A. Head, E. Glassman, G. Soares, R. Suzuki, L. Figueredo, L. D'Antoni, and B. Hartmann, "Writing reusable code feedback at scale with mixed-initiative program synthesis," in *Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale*, 2017, pp. 89–98.

[35] A. Nguyen, C. Piech, J. Huang, and L. Guibas, "Codewebs: scalable homework search for massive open online programming courses," in *Proceedings of the 23rd international conference on World wide web*, 2014, pp. 491–502.

[36] D. Galan, R. Heradio, H. Vargas, I. Abad, and J. A. Cerrada, "Automated assessment of computer programming practices: the 8-years uned experience," *IEEE Access*, vol. 7, 2019.

[37] S. N. Liao, S. Valstar, K. Thai, C. Alvarado, D. Zingaro, W. G. Griswold, and L. Porter, "Behaviors of higher and lower performing students in CS1," in *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, 2019, pp. 196–202.

[38] L. Porter and B. Simon, "Retaining nearly one-third more majors with a trio of instructional best practices in CS1," in *Proceeding of the 44th ACM technical symposium on Computer science education*, 2013.

[39] A. Vihavainen, T. Vikberg, M. Luukkainen, and J. Kurhila, "Massive increase in eager TAs: Experiences from extreme apprenticeship-based CS1," in *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, 2013, pp. 123–128.

[40] H. Kwik, H. Zhang, and E. O'Rourke, "How do students seek help and how do TAs respond? investigating help-seeking strategies in CS1 office hours," in *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2*, 2022, pp. 1130–1130.

[41] A. M. Ryan, M. H. Gheen, and C. Midgley, "Why do some students avoid asking for help? an examination of the interplay among students' academic efficacy, teachers' social–emotional role, and the classroom goal structure." *J. of educational psychology*, vol. 90, no. 3, 1998.

[42] A. Hellas, J. Leinonen, and P. Ihantola, "Plagiarism in take-home exams: help-seeking, collaboration, and systematic cheating," in *Proceedings of the 2017 ACM conference on innovation and technology in computer science education*, 2017, pp. 238–243.

[43] P. Kinnunen and B. Simon, "CS majors' self-efficacy perceptions in CS1: results in light of social cognitive theory," in *Proceedings of the seventh international workshop on Computing education research*, 2011.

[44] S. Muller, M. Babes-Vroman, M. Emenike, and T. D. Nguyen, "Exploring novice programmers' homework practices: Initial observations of information seeking behaviors," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 2020.

[45] T. W. Price, Z. Liu, V. Cateté, and T. Barnes, "Factors influencing students' help-seeking behavior while programming with human and computer tutors," in *Proceedings of the 2017 ACM Conference on international computing education research*, 2017, pp. 127–135.

[46] R. Garcia, K. Falkner, and R. Vivian, "Systematic literature review: Self-regulated learning strategies using e-learning tools for computer science," *Computers & Education*, vol. 123, pp. 150–163, 2018.

[47] A. Cook, A. Zaman, E. Hicks, K. Malasri, and V. Phan, "Try that again! how a second attempt on in-class coding problems benefits students in CS1," in *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*, 2022, pp. 509–515.

[48] S. Marwan, A. Dombe, and T. W. Price, "Unproductive help-seeking in programming: What it is and how to address it," in *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, 2020, pp. 54–60.

[49] N. M. Webb and A. Mastergeorge, "Promoting effective helping behavior in peer-directed groups," *International journal of educational research*, vol. 39, no. 1-2, pp. 73–97, 2003.

[50] C. E. O'malley, "Helping users help themselves." 1986.

[51] G. E. Xun and S. M. Land, "A conceptual framework for scaffolding iii-structured problem-solving processes using question prompts and peer interactions," *Educational Technology Research and Development*, vol. 52, no. 2, pp. 5–22, Jun 2004.

[52] T. Yan and R. Tourangeau, "Fast times and easy questions: The effects of age, experience and question complexity on web survey response times," *Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition*, vol. 22, no. 1, 2008.

[53] C. Cook, F. Heath, and R. L. Thompson, "A meta-analysis of response rates in web-or internet-based surveys," *Educational and psychological measurement*, vol. 60, no. 6, pp. 821–836, 2000.

[54] D. Thomas and A. Hunt, *The Pragmatic Programmer: your journeyman to master*. Addison-Wesley Professional, 1999.

[55] S. Kalyuga, P. Ayres, P. Chandler, and J. Sweller, "The expertise reversal effect," *Educational Psychologist*, vol. 38, no. 1, pp. 23–31, 2003.

[56] K. Ilves, J. Leinonen, and A. Hellas, "Supporting self-regulated learning with visualizations in online learning environments," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 2018, pp. 257–262.